Error-driven versus *batch* models of the acquisition of phonotactics: David defeats Goliath*

Giorgio Magri
CNRS, UiL-OTS

Nine-month-old infants already react differently to licit versus illicit sound combinations (Jusczyk et al. 1993). They thus display knowledge of the target adult phonotactics already at an early stage, when morphology is plausibly still lagging behind, hiding phonological alternations. In other words, nine-month-olds are *pure phonotactic learners* (Hayes 2004): based only on a stream of phonotactically licit forms, they manage to acquire a grammar which captures the adult phonotactics, namely is *consistent* with any licit forms and furthermore *restrictive* enough to rule out any illicit forms. How do children succeed at this learning task? This paper compares two different models of pure phonotactic learning within *Optimality Theory* (OT).

According to the *error-driven* model, the learner starts from the most restrictive initial ranking, is trained on a sequence of licit forms, and slightly re-ranks the constraints whenever the current piece of data is incorrectly predicted to be illicit by the current ranking (Tesar & Smolensky 1998; Boersma 1998; Magri 2012). Although error-driven learning has been endorsed by the acquisition literature at least since Wexler & Culicover (1980), the model seems weak from a learnability perspective. In fact, the error-driven learner never has access to the entire batch of data: re-rankings are performed instantaneously, based on the current piece of data, without ever being able to compute the implications of the current re-ranking for the rest of the data. Furthermore, the only provision towards restrictiveness is the restrictive initial ranking, which could nonetheless be unduly relaxed by the sequence of re-rankings, leading to an unrestrictive final ranking.

Error driven learning has thus been dismissed by the computational OT literature as a leaf in the wind of data, with little guarantees about the quality of the final ranking, and in particular its restrictiveness. Indeed, it has been proposed that "a ranking algorithm is unlikely to succeed simply by using a [restrictive] starting point" (Hayes 2004:p. 177). This computational literature has focused instead on the complementary class of *batch* models, such as Hayes's (2004) *Low Faithfulness Constraint Demotion* (LFCD) and Prince & Tesar's (2004) *Biased Constraint Demotion* (BCD). These algorithms are allowed to glimpse at the entire set of data and can thus better evaluate the implications of their actions. Furthermore, batch models can be endowed with aggressive biases towards restrictiveness, stated in terms of the entire batch of data.

This paper compares the error-driven and batch approaches to restrictiveness on two test cases. Section 1 briefly recalls the LFCD and BCD implementations of the batch approach and sketches an implementation of the error-driven approach, namely the *calibrated* error-driven ranking algorithm (CEDRA; Magri 2012). Section 2 introduces two test cases, namely Prince & Tesar's (2004) *Azba* typology and Hayes's (2004) *voice*×*aspiration* typology. Sections 3-5 go through the roughly seventy languages generated by these two typologies. They show that BCD and LFCD fail at restrictiveness on a number of languages in both typologies: they learn rankings which incorrectly predict too many forms to be licit. Despite the fact that error-driven learning has been portrayed in the literature as computationally weaker, the CEDRA surprisingly succeeds on each single language in both typologies, at least in the idealized regime of uniform sampling (i.e., all training data have the same frequency). In other words, David defeats Goliath. Section 6 concludes the paper. Magri (2014) offers a more detailed discussion of the simulation results presented in this paper.

1 Batch (LFCD, BCD) and error-driven (CEDRA) pure phonotactic learners

A pure phonotactic learner is trained on licit surface forms (e.g. [d^{h}a]). Any such form is interpreted by the learner as the faithful realization of that same form construed as an underlying form (e.g., /d^{h}a/; Tesar

* The research reported in this paper has been supported in part by a Marie Curie Fellowship for Career Development.

Proceedings of *Phonology 2013*
Completed February 18, 2014

2013). The learning data thus consist of triplets of an underlying form, the faithful winner form, and all possible non-faithful loser forms (e.g., ~~[da]~~, ~~[tʰa]~~, ...; loser forms are stricken out), as in (1).

(1) $\big(/\text{d}^\text{h}\text{a}/, [\text{d}^\text{h}\text{a}], \sout{[\text{da}]}\big) = \begin{bmatrix} C_1 & C_2 & \dots & C_n \\ \text{W} & e & \dots & \text{L} \end{bmatrix}$

One such triplet is completely described by its corresponding *Elementary Ranking Condition* (ERC; Prince 2002), namely a tuple with an entry for each constraint, equal to one of the three symbols W, L, or e depending on whether the corresponding constraint is *winner-* or *loser-preferring* or *even* relative to the underlying/winner/loser form triplet represented by that ERC. The data set can thus equivalently be represented as a set of ERCs, which I will stack one on top of the other, into an ERC matrix.

Given a set of ERCs corresponding to n constraints, the *Constraint Demotion* (CD) algorithm (Tesar & Smolensky 1998) constructs a ranking iteratively as follows. At the first iteration, CD looks for a constraint which is not loser-preferring relative to any ERC; if there is more than one such constraint, it chooses one at random – call it C_1; it then ranks that constraint at the top; and it simplifies the set of ERCs by deleting any ERC where that constraint C_1 is winner-preferring. At the second iteration, CD looks for a constraint (different from C_1) which is never loser-preferring relative to the updated set of ERCs; if there is more than one such constraint, it chooses one at random – call it C_2; it then ranks that constraint C_2 right underneath C_1; and it further simplifies the current set of ERCs by deleting any ERC where C_2 is winner-preferring. And so on: at each iteration, CD chooses a suitable constraint, it ranks that constraint at the top available slot, and it simplifies the current set of ERCs. After n iterations, all ERCs have been deleted and all constraints have been ranked. The ranking $C_1 \gg C_2 \gg \dots$ assembled by CD is guaranteed to be consistent with each ERC in the data set. If the ERCs in that set represent the underlying/winner/loser form triplets corresponding to all possible licit winner forms, their faithful underlying forms and all possible unfaithful loser forms, then the ranking assembled by CD correctly predicts each training form to be indeed licit, namely faithfully realized rather than reduced to some non-faithful candidate.

The ranking assembled by CD could nonetheless fail at restrictiveness, namely it could predict too many forms to be licit. This means that at some iterations, CD was given a choice among multiple constraints (i.e., multiple constraints were not loser-preferring relative to the current set of ERCs) and made a bad choice. In order to bias CD towards a restrictive ranking, the algorithm needs to choose carefully, namely not a random but according to proper *choice criteria* aimed at restrictiveness. LFCD and BCD are two ways of enriching CD with choice criteria to be used at iterations where multiple constraints are available to be ranked.

To start, since markedness constraints work towards restrictiveness, they should be ranked as high as possible. In the logics of CD (which assembles the final ranking from the top down), this means that markedness constraints should be chosen as soon as possible, leading to the choice criterion in (2).

(2) Choose a markedness instead of a faithfulness constraint whenever possible, namely at any iteration where there is a markedness constraint which is not loser-preferring relative to the current set of ERCs.

If, instead, all markedness constraints are loser-preferring at the current iteration, the algorithm needs to choose a faithfulness constraint. In the test cases described in section 2, faithfulness constraints come in two varieties: *general* and *positional*. Let's say that a faithfulness constraint is *specific* at the current iteration iff it is positional or else it is general but its corresponding positional companion has already been ranked at an earlier iteration (Hayes 2004). Since faithfulness constraints work against restrictiveness, their effect should be minimized. A positional faithfulness constraint by definition has a more limited effect than its corresponding general companion. This suggests that a positional faithfulness constraint should be ranked above the corresponding general one. In the logics of CD, this means that the choice should always be restricted to faithfulness constraints which are specific at the current iteration, leading to the choice criterion in (3). Note that (3) guarantees that a positional faithfulness constraint is ranked above the corresponding general faithfulness constraint. But it does not prevent a general faithfulness constraint to be ranked above a non-corresponding positional constraint.[1]

[1] The choice criterion (3) has ben suggested by Hayes (2004) for LFCD. Prince & Tesar's (2004) BCD does not adhere to this criterion (see footnote 3 below and Hayes 2004:Appendix A). In what follows, I consider instead a variant of BCD enriched with (3), as stated in my reformulation (7) of BCD's choice criterion.

(3) At any iteration where all markedness constraints are loser-preferring relative to the current set of ERCs, choose a faithfulness constraint which is specific at that iteration.

In general, multiple specific faithfulness constraints can be chosen at a given iteration. How should CD choose one? LFCD and BCD provide two different choice criteria.

Hayes's (2004) choice criterion is based on the notion of *autonomy* defined in (4). The autonomy of a constraint F relative to an ERC $\mathbf{a}$ is 0 (minimum value) if F is not winner-preferring relative to that ERC $\mathbf{a}$. Otherwise, its autonomy is set equal to the inverse of the number of constraints which are winner-preferring relative to that ERC. Thus, the autonomy of F is 1 (maximum value) if F is the unique winner-preferrer, it is $1/2$ if there is another winner-preferrer, it is 1/3 if there are two other winner-preferrers, and so on. Finally, the autonomy of F relative to multiple ERCs $\mathbf{a}_1, \mathbf{a}_2, \ldots$ is the largest autonomy of F over those ERCs.

(4) a. $autono_{\mathbf{a}}(F) = \begin{cases} 0 & \text{if } F \text{ is not winner-preferring in the ERC } \mathbf{a} \\ 1/\text{number of winner-preferrers} & \text{otherwise} \end{cases}$

b. $autono_{\mathbf{a}_1,\mathbf{a}_2,\ldots}(F) = \max\left\{autono_{\mathbf{a}_1}(F), autono_{\mathbf{a}_2}(F), \ldots\right\}$

Hayes suggests the choice criterion (5), which favors maximally autonomous specific faithfulness constraints. The intuition is that, "by only allowing maximally autonomous faithfulness constraints to be installed, we can limit installation to [faithfulness constraints] that are most likely to be truly necessary, letting the less autonomous constraints sink further down to a point where they will not lead to over-generation" (p. 181).

(5) At any iteration where all markedness constraints are loser-preferring relative to the current ERCs, choose the specific faithfulness constraint with maximum autonomy (relative to the current set of ERCs); if multiple specific faithfulness constraints tie for maximum autonomy, choose one at random.

CD modified with the two choice criteria (2) and (5) is called *Low Faithfulness* CD (LFCD).

Prince & Tesar (2004) resort to the notion of *freeing up* markedness constraints, as defined in (6). The intuition is that a set of faithfulness constraints frees up a markedness constraint provided CD will be able to choose that markedness constraint immediately after it has chosen all those faithfulness constraints.

(6) a. A set of constraints $\{F', F'', \ldots\}$ *immediately frees up* a constraint M relative to a set of ERCs iff at least one of $F', F'', \ldots$ is winner-preferring in any ERC where M is loser-preferring.

b. A set of constraints $\{F', F'', \ldots\}$ *frees up* a constraint M relative to a set of ERCs iff there exists a sequence of constraints $M^1, M^2, \ldots, M^m$ such that the set $\{F', F'', \ldots\}$ immediately frees up M^1; the set $\{F', F'', \ldots, M^1\}$ immediately frees up M^2, …, the set $\{F', F'', \ldots, M^1, \ldots, M^m\}$ immediately frees up M.

Prince and Tesar suggest the choice criteria (7), which favors specific faithfulness constraints which free up markedness constraints. In cases where there are multiple sets of such faithfulness constraints, two dimensions are relevant: what is the number of faithfulness constraints needed to free up some markedness constraints? and what is the number of markedness constraints which are freed up? They conjecture that the former dimension is more important than the latter: we should try to minimize the number of faithfulness constraints involved, even if we have to settle for a smaller number of markedness constraints freed up.

(7) At any iteration where all markedness constraints are loser-preferring relative to the current set of ERCs, choose a faithfulness constraint in the smallest set of specific faithfulness constraints which frees up markedness constraints; if there is more than one such smallest set, choose one smallest set of specific faithfulness constraints which frees up the largest number of markedness constraints.[2]

CD modified with the two choice criteria (2) and (7) is called *Biased* CD (BCD).

CD is a *batch* ranking algorithm, namely it looks at the ERC matrix "by column". In the sense that it looks for constraints whose columns contain no L's in the current ERC matrix. LFCD and BCD take advantage of this fact: the notions of autonomy and freeing-up require the algorithm to be able to glimpse at the entire ERC matrix. An *error-driven* ranking algorithm (EDRA) instead looks at the ERC matrix "by

[2] The formulation (7) says nothing about cases where the set of all specific faithfulness constraints fails at freeing up any markedness constraint. This issue never arises in the cases discussed below. See Magri 2014 for alternative formulations.

row", in the following sense. The algorithm assigns to each constraint a number, called its *ranking value*. These values implicitly define a ranking: a constraint is ranked above another constraint iff the ranking value of the former is larger than the ranking value of the latter. At every iteration, the algorithm is provided with a single piece of data, namely a single ERC (that is, a single row of the ERC matrix). If some constraint which is winner-preferring relative to that ERC has a current ranking value which is larger than the ranking value of every constraint which is instead loser-preferring, the algorithm does nothing and waits for another piece of data. Otherwise, the algorithm slightly modifies its current ranking values as in (8). Loser-preferring constraints are demoted by a small amount, say 1 for concreteness, as in (8a). Crucially, not all loser-preferring constraints are demoted, but only those that need to be demoted, namely the *undominated* ones, whose current ranking value is not smaller than the ranking value of some winner-preferring constraint. Furthermore, the winner-preferring constraints are promoted by a small *promotion amount* $p \geq 0$. This promotion amount needs to be properly *calibrated* (Magri 2012), namely needs to be strictly smaller than the inverse of the number w of winner-preferring constraints. This condition is satisfied by the choice of the promotion amount p in (8b), used in the simulations reported below.

(8) a. Decrease the ranking value of each undominated loser-preferring constraint by 1.

b. Increase the ranking value of each winner-preferring constraint by $p = \frac{1}{w+1}$.

Because of the calibrated promotion amount, the resulting algorithm is called a *calibrated* EDRA (CEDRA). The importance of the non-null promotion amount will become clear from the simulations below. Despite the non-null promotion amount, the algorithm is guaranteed to only perform a finite (and small) number of updates. In other words, if the training ERCs represent the underlying/winner/loser form triplets corresponding to all possible licit winner forms, their faithful underlying forms and all possible unfaithful loser forms, then the ranking learned by the CEDRA correctly predicts each licit form to be indeed licit.

Again as in the case of CD, the final ranking learned by the CEDRA could nonetheless fail at restrictiveness, namely it could predict too many forms to be licit. In order to bias the CEDRA towards restrictiveness, I will add two additional provisions. The first provision is that the markedness constraints are initially ranked above the faithfulness constraints, as in (9). In other words, the algorithm starts from a most restrictive grammar. Assumption (9) translates the batch choice criterion (2) into the error-driven setting.

(9) The initial ranking value of the markedness constraints is larger than the initial ranking value of the faithfulness constraints.

The second provision towards restrictiveness is that a positional faithfulness constraint is initially ranked above the corresponding general faithfulness constraint, as stated in (10a); and that ranking configuration is actively enforced throughout learning, by avoiding promoting a general faithfulness constraint above the corresponding positional one, as stated in (10b). In other words, the algorithm never considers rankings where a general faithfulness constraint is ranked above the corresponding positional one. Note that condition (10) does not prevent the CEDRA to promote a general faithfulness constraint above a non-corresponding positional faithfulness constraint. Assumption (10) translates the batch choice criterion (3) into the error-driven setting. In the simulations reported in this paper, the initial ranking values are: 3000 for the markedness constraints; 700 for the positional faithfulness constraints; 0 for the general faithfulness constraints.

(10) a. The initial ranking value of the positional faithfulness constraints is larger than the initial ranking value of the general faithfulness constraints.

b. Whenever an update would promote a general faithfulness constraint above the corresponding positional faithfulness constraint, avoid performing that constraint promotion.

The final grammar learned by an error-driven learner might in principle depend on the relative frequencies with which the data are sampled and fed to the algorithm. The simulations reported below assume an idealized uniform sampling regime, whereby all licit forms are sampled and fed to the CEDRA with equal probability. Repeated simulations always yielded identical results.

2 Two test cases

This paper tests LFCD, BCD, and the CEDRA on two test cases. The first test case is defined through the set of forms and the set of constraints in (11); the generating function only modifies obstruent voicing,

and is omitted for brevity (Prince & Tesar 2004, based on Lombardi 1999). These typological specifications are centered around the two features [STOP-VOICING] and [FRICATIVE-VOICING]. These two features are associated with general faithfulness constraints F_1, F_2, faithfulness constraints $F_1^{\text{pos}}, F_2^{\text{pos}}$ specialized for the onset position, and markedness constraints M_1, M_2. The two features interact through the markedness constraint M which bans certain combinations of their values (namely, bans sequences of stops and fricatives that disagree in voicing). I will refer to this test case as the *Azba typology*.

(11)
$$\left\{\begin{array}{cccc} \text{pa} & \text{ba} & \text{ap} & \text{ab} \\ \text{sa} & \text{za} & \text{as} & \text{az} \\ \text{apsa} & \text{apza} & \text{absa} & \text{abza} \\ \text{aspa} & \text{azpa} & \text{asba} & \text{azba} \end{array}\right\} \quad \left\{\begin{array}{ll} F_1 = \text{ID[STOP-VOICE]} & F_2 = \text{ID[FRIC-VOICE]} \\ F_1^{\text{pos}} = \text{ID[STOP-VOICE]/\_V} & F_2^{\text{pos}} = \text{ID[FRIC-VOICE]/\_V} \\ M_1 = {}^*\text{[+STOP-VOICE]} & M_2 = {}^*\text{[+FRIC-VOICE]} \\ \multicolumn{2}{c}{M = \text{AGREE[VOICE]}} \end{array}\right\}$$

This Azba typology (computed using `OTsoft`; Hayes et al. 2003) consists of 41 OT grammars, corresponding to 37 different languages (some grammars generate the same language). Two of these languages are trivial, namely the one which consists of all sixteen forms (the largest language in the typology) and the one which only consists of the six unmarked forms [pa], [ap], [sa], [as], [apsa], and [aspa] (the smallest language in the typology). Some pairs of languages are formally equivalent due to the symmetry between the two features [STOP-VOICING] and [FRICATIVE-VOICING] (see Magri 2014 for details). Because of these equivalences, we only need to consider 19 of the 35 non-trivial languages.

The second test case considered in this paper is defined through the set of forms and the set of constraints in (12); the generating function only modifies obstruent voicing and aspiration, and is omitted for brevity (Hayes 2004). These typological specifications are centered around the two features [VOICE] and [SPREAD GLOTTIS]. These two features are associated with general faithfulness constraints F_1, F_2, faithfulness constraints F_1^{pos}, F_2^{pos} specialized for the onset position, and markedness constraints M_1, M_2 and M_3.The two features interact through the markedness constraint M which bans certain combinations of their values (i.e., bans breathy voice). I will refer to this test case as the *voicing×aspiration typology*.

(12)
$$\left\{\begin{array}{cccc} \text{ta}, & \text{da}, & \text{t}^{\text{h}}\text{a}, & \text{d}^{\text{h}}\text{a}, \\ \text{at}, & \text{ad}, & \text{at}^{\text{h}}, & \text{ad}^{\text{h}}, \\ \text{ata}, & \text{ada}, & \text{at}^{\text{h}}\text{a}, & \text{ad}^{\text{h}}\text{a} \end{array}\right\} \quad \left\{\begin{array}{ll} F_1 = \text{IDENT[VOICE]} & F_2 = \text{IDENT[SPREAD]} \\ F_1^{\text{pos}} = \text{IDENT[VOICE]/\_V} & F_2^{\text{pos}} = \text{IDENT[SPREAD]/\_V} \\ M_1{}^*\text{[+VOICE, -SON]} & M_2 = {}^*\text{[+SPREAD]} \\ \overline{M}_1 = {}^*\text{[}-\text{VOICE, }-\text{SON]/V\_V} & \\ \multicolumn{2}{c}{M = {}^*\text{[+SPREAD, +VOICE]}} \end{array}\right\}$$

This typology consists of 56 OT grammars, corresponding to 40 different languages. One of these 40 languages is trivial, namely the largest language which contains all sixteen forms. There is no single smallest language consisting of only unmarked forms, because all forms with an intervocalic stop violate some markedness constraint. I focus on the 39 non-trivial languages.

It is convenient to sort the languages in these two typologies into five types I-V, based on the "type" of ranking conditions among faithfulness constraints they "require". Sections 3-5 will then look at each of the five types of languages in turn. Here is how this taxonomy of languages is defined. To start, let me distinguish four types of ranking conditions among faithfulness constraints, listed in (13).

(13) a. $F_1^{\text{pos}} \gg F_1$, $F_2^{\text{pos}} \gg F_2$ b. $F_1^{\text{pos}} \gg F_2$, $F_2^{\text{pos}} \gg F_1$ c. $F_1^{\text{pos}} \gg F_2^{\text{pos}}$, $F_2^{\text{pos}} \gg F_1^{\text{pos}}$, $F_1 \gg F_2$, $F_2 \gg F_1$ d. $F_2 \gg F_1^{\text{pos}}$, $F_1 \gg F_2^{\text{pos}}$

The first type (13a) of ranking conditions require a positional faithfulness constraint to be ranked above the corresponding general faithfulness constraint. These ranking conditions are trivial because they don't need to be learned: they are enforced through condition (3) in the batch case and through condition (10) in the error-driven case. The second type (13b) of ranking conditions require a positional faithfulness constraint to be ranked above a non-corresponding general faithfulness constraint. These ranking conditions need to be learned: they cannot be enforced, as the target language might in principle require the reverse ranking condition, as in (13d). Yet, the privileged status accorded to positional over general faithfulness constraints in order to enforce the ranking conditions of type (13a), might also be of some help with these ranking conditions of type (13b). For instance, the fact that all positional faithfulness constraints start out initially

ranked above all general faithfulness constraints in the error-driven setting, gives a boost to the positional faithfulness constraints which might be helpful also in learning these ranking conditions of type (13b). The third type (13c) of ranking conditions require a positional (or a general) faithfulness constraint to be ranked above another positional (or another general, respectively) faithfulness constraint. These ranking conditions are intuitively harder to learn, as the two constraints which need to be ranked relative to each other are on a par with respect to the a priori biases built into the algorithms. Finally, the fourth type (13d) of ranking conditions require a general faithfulness constraint to be ranked above a (non-corresponding) positional faithfulness constraint. These ranking conditions are intuitively the hardest to learn, as they require the algorithm to be able able to actively overcome the indiscriminate preference for positional above general faithfulness constraints built into the notion (3) of specificity or the choice (10a) of the initial ranking values.

A *total* ranking ranks any two faithfulness constraints relative to each other. It is thus unsuitable to model the intuition that a certain ranking enforces only certain types of ranking conditions among the faithfulness constraints. Thus, let me make use of the notion of a *partial ranking*, namely a possibly non-total order over the constraints. Partial rankings can be classified into five types, based on the types (13) of ranking conditions they impose among the faithfulness constraints. A partial ranking is called of *type I* provided it does not rank any two faithfulness constraints relative to each other. It is called of *type II* provided it only enforces ranking conditions of type (13a) among the faithfulness constraints, but not of any other type (13b)-(13d). It is called of *type III* provided it enforces ranking conditions of type (13b) and possibly (13a), but not of types (13c) and (13d). A partial ranking is called of *type IV* provided it enforces ranking conditions of type (13c) and possibly (13a) and (13b), but not of type (13d). Finally, a partial ranking is called of *type V* provided it enforces ranking conditions of type (13d) and possibly also of types (13a)- (13c) among the faithfulness constraints.

A partial ranking is said to *generate* a certain language provided each one of its total refinements generates that language in the usual OT sense (Yanovich 2012). The languages in a typology can now be sorted into five types depending on the types of ranking conditions among faithfulness constraints they require. More explicitly, a language is of *type I* provided it does not require any faithfulness constraint to be ranked relative to any other faithfulness constraint, in the sense that it can be generated by a partial ranking of type I. A language is called of *type II* provided it requires a positional faithfulness constraint to be ranked above the corresponding general faithfulness constraint, in the sense that it can be generated by a partial ranking of type II but not by any partial raking of type I. A language is called of *type III* provided it requires a positional faithfulness constraint to be ranked above a non-corresponding general faithfulness constraint, in the sense that it can be generated by a partial ranking of type III but not by any partial ranking of types I or II. A language is called of *type IV* provided it requires two positional or two general faithfulness constraints to be ranked relative to each other, in the sense that it can be generated by a partial ranking of type IV but not by any partial rankings of types I-III. Finally, a language is called of *type V* provided it requires a general faithfulness constraint to be ranked above a (non-corresponding) positional faithfulness constraint, in the sense that it can only be generated by a partial ranking of type V. In the rest of this paper, I test the performance of the three learners LFCD, BCD and the CEDRA on languages of different types in the two typologies (11) and (12).

3 Languages of type I and II

Recall that a language is of type I provided it does not require any relative ranking among the faithfulness constraints, not even between a positional and the corresponding general faithfulness constraint. Among the 19 non-redundant languages in the Azba typology, there are three languages of type I; in the voicing$\times$aspiration typology, there are ten languages of type I. As expected, all three algorithms LFCD, BCD, and the CEDRA succeed at learning a restrictive ranking on each of these languages of type I (Magri 2014). The case of languages of type II is analogous. Recall that a language is of type II provided it only requires a positional faithfulness constraint to be ranked above the corresponding general faithfulness constraint as in (13a), but does not require any other types of ranking conditions among the faithfulness constraints. Among the 19 non-redundant languages in the Azba typology, there are five languages of type II; in the voicing$\times$aspiration typology, there are 19 languages of type II. All three algorithms LFCD, BCD, and the CEDRA succeed at learning a restrictive ranking on these languages of type II. This is because all three algorithms have been endowed with a bias which keeps a positional faithfulness constraint above the corresponding general one, namely the preference (3) for specific constraints in the case of LFCD and BCD and the clause (10) in the case of the CEDRA. The algorithms then manage to intersperse the markedness

constraints correctly into that hierarchy of positional-above-general faithfulness constraints (Magri 2014), leading to a restrictive ranking.[3]

4 Languages of type III and V

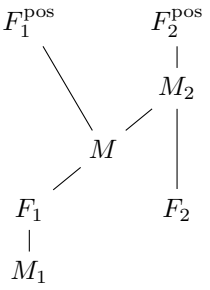
Recall that a language is of type III provided it requires a positional faithfulness constraint to be ranked above a non-corresponding general faithfulness constraint as in (13b) – as well as, possibly, above the corresponding general faithfulness constraint, as in (13a). There are no languages of type III in the voicing×aspiration typology; among the 19 non-redundant languages in the Azba typology, there are four languages of type III. Recall next that a language is of type V provided it requires a general faithfulness constraint to be ranked above a non-corresponding positional faithfulness constraint, as in (13d). There are no languages of type V in the voicing×aspiration typology; among the 19 non-redundant languages in the Azba typology, there is only one language of type V. Since the ranking conditions of type (13b) and (13d) required by languages of type III and V are one the reverse of the other, it is useful to consider these two types of languages together.

LFCD fails on two of the four languages of type III in the Azba typology. Since the reason of the failure is identical in the two languages, here I focus on one of them. Consider the language which only consists of the forms [ba], [ab], [za], [abza] and [asba] (plus the unmarked forms). The corresponding ERC set is provided in (14a). This language is generated by a total ranking iff it is a refinement of the partial ranking in (14b). The latter ranks the positional faithfulness constraint F_2^{pos} above the non-corresponding general faithfulness constraint F_1, certifying that this language is indeed of type III.

(14) a.

	F_1	F_2	F_1^{pos}	F_2^{pos}	M	M_1	M_2
(/ba/, [ba], ~~[pa]~~)	W		W			L	
(/ab/, [ab], ~~[ap]~~)	W					L	
(/za/, [za], ~~[sa]~~)		W		W			L
(/abza/, [abza], ~~[apza]~~)	W				W	L	
(/abza/, [abza], ~~[absa]~~)		W		W	W		L
(/abza/, [abza], ~~[apsa]~~)	W	W		W		L	L
(/asba/, [asba], ~~[aspa]~~)	W		W		L	L	
(/asba/, [asba], ~~[azba]~~)		W			L		W
(/asba/, [asba], ~~[azpa]~~)	W	W	W			L	W

b.



Already at the first iteration, LFCD needs to choose a faithfulness constraint, because all markedness constraints are loser-preferring. By specificity, the choice boils down to F_1^{pos} and F_2^{pos}. Both of them are winner-preferring in ERCs with only one other winner-preferring constraint. LFCD will thus choose at random. Suppose it chooses F_1^{pos}. At the second iteration, another faithfulness constraint needs to be chosen. By specificity, the choice boils down to F_1 (which is general and yet specific, because its positional mate has already been ranked) and F_2^{pos}. LFCD chooses the former, because it has maximum autonomy, as it is winner-preferring in the ERC (/ab/, [ab], ~~[ap]~~) where there are no other winner-preferring constraints. The algorithm thus fails to learn that F_2^{pos} needs to be ranked above F_1.

An easy fix would be to tighten the definition of specificity. According to the current definition, a general faithfulness constraint counts as specific as soon as its positional companion has been ranked. In the case of (14), this loose condition allows for F_1 to count as specific too early. One might thus tighten the notion of specificity, requiring that *all* positional faithfulness constraints have been ranked before any general faithfulness constraint can be chosen. This variant would ensure that LFCD successfully always ranks the two positional faithfulness constraints above the two general faithfulness constraints. This move is successful in the case of a language of type III such as (14). Yet, it would lead into troubles with languages of type V, which require a general faithfulness constraint to be ranked above a non-corresponding positional one.

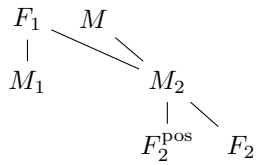
Indeed, consider the language in the Azba typology which consists of [ba], [ab], [abza] and [azba] (plus the unmarked forms). The corresponding ERC set is provided in (15a). This language is generated by a total ranking iff it is a refinement of the partial ranking (15b). The latter ranks the general faithfulness constraint F_1 above the non-corresponding positional constraint F_2^{pos}, certifying that this language is indeed of type V.

[3] Languages of type II can be used to motivate the extension of Hayes' specifity preference (3) from LFCD to BCD (see footnote 1). For instance, BCD without specificity would fail on all five languages of type II in the Azba typology.

(15) a.

	F_1	F_2	F_1^{pos}	F_2^{pos}	M	M_1	M_2
(/ba/, [ba], ~~[pa]~~)	W		W			L	
(/ab/, [ab], ~~[ap]~~)	W					L	
(/abza/, [abza], ~~[apza]~~)	W				W	L	
(/abza/, [abza], ~~[absa]~~)		W		W	W		L
(/abza/, [abza], ~~[apsa]~~)	W	W		W		L	L
(/azba/, [azba], ~~[azpa]~~)	W		W		W	L	
(/azba/, [azba], ~~[asba]~~)		W			W		L
(/azba/, [azba], ~~[aspa]~~)	W	W	W			L	L

b.



LFCD with the original notion of specificity succeeds on this language. At the first iteration, it chooses the markedness constraint M and simplifies the ERC set accordingly. At the next iteration, it has to choose a faithfulness constraint. Because of specificity, the choice boils down to F_1^{pos} and F_2^{pos}. It chooses F_1^{pos}, because it is winner-preferring in the ERC (/ba/, [ba], ~~[pa]~~) which has only one additional winner-preferrer, while F_2^{pos} is only winner-preferring in the ERC (/abza/, [abza], ~~[apsa]~~), which has two additional winner-preferrers. At the next iteration, LFCD has to choose another faithfulness constraint. Because of Hayes' original definition of specificity, F_1 counts as specific although general, because its companion positional constraint F_1^{pos} has been ranked already. The choice thus boils down to the two specific constraints F_1 and F_2^{pos}. LFCD correctly chooses F_1, because it is winner-preferring in the ERC (/ab/, [ab], ~~[ap]~~) which has no additional winner-preferring constraints. LFCD with the original definition of specificity thus manages to choose the general faithfulness constraint F_1 before the positional faithfulness constraint F_2^{pos}, thus ensuring that the former ends up above the latter. This success would be lost if Hayes' original notion of specificity were to be modified as suggested above, forcing all positional faithfulness constraints to be ranked before and thus above all general faithfulness constraints.

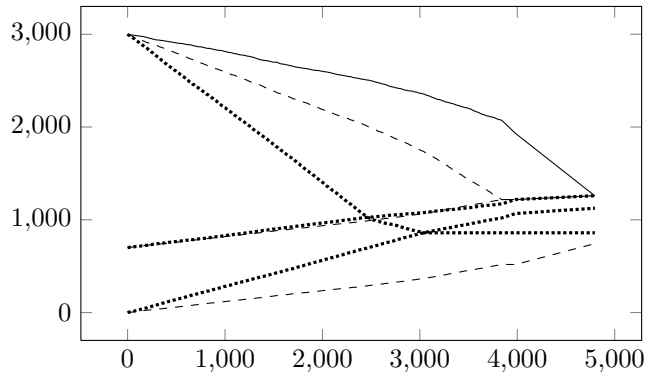
BCD exemplifies this same tension between languages of type III and type V, but with the opposite performance: it succeeds on all four languages of type III in the Azba typology, such as the one in (14); but at the price of failing on the language of type V described in (15). To see how BCD succeeds on languages of type III, consider again the case of language (14). Already at the first iteration, BCD needs to choose a set of specific faithfulness constraints which free up some markedness constraints. Neither F_1^{pos} nor F_2^{pos} individually free up any markedness constraint. Thus, the smallest set of specific faithfulness constraints required to free up some markedness constraint consists of both of them. BCD will thus chose these two constraints at the first and second iteration, yielding the correct final ranking. BCD's tendency to produce final rankings where both positional faithfulness constraints are ranked above both general faithfulness constraints is of course an advantage in the case of the four languages of type III. Yet, it turns into a disadvantage in the case of languages of type V such as (15). When run on the latter language, BCD chooses M at the first iteration, and simplifies the ERC set accordingly. At the next iteration, the two general faithfulness constraints F_1 and F_2 each individually would suffice to free up the corresponding markedness constraint. Yet, I am assuming that BCD adheres to specificity – an assumption which has proven crucial to guarantee its success on languages of type II (see footnote 3). Because of specificity, BCD has first to check whether it can construct a set of specific faithfulness constraints which free up some markedness constraints. The two positional faithfulness constraints F_1^{pos} and F_2^{pos} individually do not suffice to free up any markedness constraint. Yet, the set consisting of both of them together suffices to free up M_2. By specificity, BCD will thus choose these two positional faithfulness constraints at the second and third iteration. The algorithm will thus return an unrestrictive final ranking, which incorrectly ranks F_2^{pos} above F_1.

In conclusion, there is a tension between languages of types III and V and these batch algorithms are not versatile enough to resolve this tension. The situation is different in the case of the CEDRA. In (16a) and (17a), I provide the final ranking values learned by the CEDRA when trained on the two languages (14a) and (15a). These final ranking values enforce all the required ranking conditions (14b) and (15b), showing that the CEDRA has succeeded on both test cases. The initial distance between the positional and the general faithfulness constraints is large enough that the algorithm learns to keep a positional above a non-corresponding general faithfulness constraint, when that is needed as in the case of (14) and the other three languages of type III in the Azba typology. But not so large that the algorithm fails at learning the reverse ranking of a general above a non-corresponding positional faithfulness constraints, when that is needed as in the case of the language (15) of type V.

(16) a.

F_2^{pos}	1257.8
F_1^{pos}	1257.3
M_2	1256.9
M	1256.0
F_1	1121.8
M_1	862.0
F_2	729.8

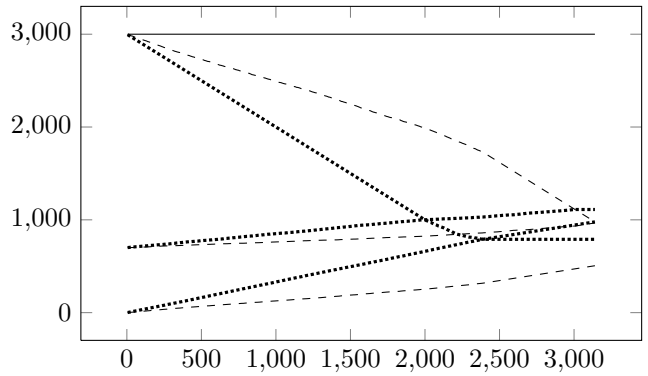
b.



(17) a.

M	3000.0
F_1^{pos}	1105.0
F_1	980.5
M_2	980.0
F_2^{pos}	965.0
M_1	795.0
F_2	505.0

b.



The diagrams (16b) and (17b) plot the corresponding ranking dynamics, namely the ranking values (vertical axis) as a function of time (horizontal axis). The ranking values of the constraints F_1, F_1^{pos}, and M_1 are plotted with dotted lines (the three constraints can be distinguished because of their initial ranking values); the ranking values of the constraints F_2, F_2^{pos}, and M_2 are plotted with dashed lines; the ranking value of the constraint M is plotted with a solid line. See Magri (2014) for an analysis of these two ranking dynamics, and thus an explanation of how the CEDRA resolves the tension between languages of types III and V. Here, I only want to underscore the fact that this success would have been impossible if the promotion amount had been set equal to zero, as in Tesar & Smolensky's (1998) original demotion-only EDRA. If the algorithm performs no constraint promotion, then it never re-ranks the faithfulness constraints and thus will never be able to distinguish between the two cases (14) and (15) which require the opposite relative ranking of two faithfulness constraints.

5 Languages of type IV

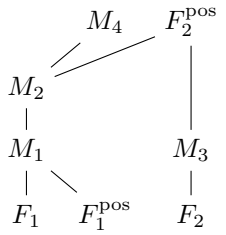
Recall that a language is of type IV provided it requires a positional (or a general) faithfulness constraint to be ranked relative to another positional (respectively, another general) faithfulness constraint. Among the 19 non-redundant languages in the Azba typology, there are six languages of type IV. BCD fails on one of them, LFCD fails on three of them. Both are thus outperformed by the CEDRA, which succeeds on all six languages. For details about the performance of these three algorithms on the languages of type IV in the Azba typology, see Magri (2014). In the rest of this section, I focus on the voicing×aspiration typology, which contains ten languages of type IV.

LFCD succeeds on only two of these ten languages of type IV. Furthermore, a closer look at LFCD's failure on the remaining eight languages reveals that its success on two languages is somewhat "fortuitous". To illustrate this point, consider the language which only consists of [t$^{\text{h}}$a], [ada], and [at$^{\text{h}}$a] (plus the unmarked forms), capturing the pattern of voicing and aspiration in Korean. The ERC set corresponding to this language is provided in (18a). This language is generated by a total ranking iff it is a refinement of the partial ranking (18b). The latter crucially requires the positional faithfulness constraint F_2^{pos} to be ranked above the positional constraint F_1^{pos}, certifying that this language is of type IV. This is indeed the case used in Hayes (2004) to motivate and illustrate the mechanics of LFCD.

(18) a.

	F_1	F_2	F_1^{pos}	F_2^{pos}	M_1	M_2	M_3	M_4
(/t$^{\text{h}}$a/, [t$^{\text{h}}$a], ~~[ta]~~)		W		W			L	
(/t$^{\text{h}}$a/, [t$^{\text{h}}$a], ~~[da]~~)	W	W	W	W	W		L	
(/t$^{\text{h}}$a/, [t$^{\text{h}}$a], ~~[d$^{\text{h}}$a]~~)	W		W		W			W
(/ada/, [ada], ~~[ata]~~)	W		W		L	W		
(/ada/, [ada], ~~[at$^{\text{h}}$a]~~)	W	W	W	W	L	W	W	
(/ada/, [ada], ~~[ad$^{\text{h}}$a]~~)		W		W			W	W
(/at$^{\text{h}}$a/, [at$^{\text{h}}$a], ~~[ata]~~)		W		W			L	
(/at$^{\text{h}}$a/, [at$^{\text{h}}$a], ~~[ada]~~)	W	W	W	W	W	L	L	
(/at$^{\text{h}}$a/, [at$^{\text{h}}$a], ~~[ad$^{\text{h}}$a]~~)	W		W		W	L		W

b.



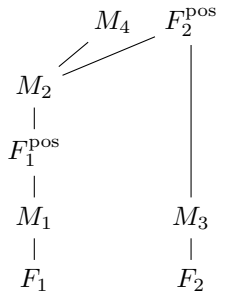
At the first iteration, LFCD chooses the markedness constraint M_4 and simplifies the ERC set accordingly. At the second iteration, it has to choose a faithfulness constraint. By specificity, the choice boils down to one of the two positional faithfulness constraints F_1^{pos} and F_{pos}. By autonomy, LFCD chooses F_2^{pos}, because it is winner-preferring in the ERC (/t$^{\text{h}}$a/, [t$^{\text{h}}$a], ~~[ta]~~) which has a unique additional winner-preferrer, while every ERC where F_1^{pos} is winner-preferring has two or more additional winner-preferrers. Since F_2^{pos} is more autonomous and thus chosen before F_1^{pos}, LFCD ends up correctly ranking the former above the latter.

Yet, it turns out that LFCD has succeeded on the test case (18) somewhat by accident. In fact, consider a language which only differs because it deems [da] licit as well. The corresponding ERC set is provided in (19a). This language is generated by a total ranking iff it is a refinement of the partial ranking (19b). The latter ranking is a small variant of the one in (18b): they only differ in their bottom portion, for the relative ranking of the two constraints F_1^{pos} and M_1.

(19) a.

	F_1	F_2	F_1^{pos}	F_2^{pos}	M_1	M_2	M_3	M_4
(/da/, [da], ~~[ta]~~)	W		W		L			
(/da/, [da], ~~[t$^{\text{h}}$a]~~)	W	W	W	W	L		W	
(/da/, [da], ~~[d$^{\text{h}}$a]~~)		W		W			W	W
(/t$^{\text{h}}$a/, [t$^{\text{h}}$a], ~~[ta]~~)		W		W			L	
(/t$^{\text{h}}$a/, [t$^{\text{h}}$a], ~~[da]~~)	W	W	W	W	W		L	
(/t$^{\text{h}}$a/, [t$^{\text{h}}$a], ~~[d$^{\text{h}}$a]~~)	W		W		W			W
(/ada/, [ada], ~~[ata]~~)	W		W		L	W		
(/ada/, [ada], ~~[at$^{\text{h}}$a]~~)	W	W	W	W	L	W	W	
(/ada/, [ada], ~~[ad$^{\text{h}}$a]~~)		W		W			W	W
(/at$^{\text{h}}$a/, [at$^{\text{h}}$a], ~~[ata]~~)		W		W			L	
(/at$^{\text{h}}$a/, [at$^{\text{h}}$a], ~~[ada]~~)	W	W	W	W	W	L	L	
(/at$^{\text{h}}$a/, [at$^{\text{h}}$a], ~~[ad$^{\text{h}}$a]~~)	W		W		W	L		W

b.



At the first iteration, LFCD chooses the markedness constraint M_4 and simplifies the ERC set accordingly. At the second iteration, LFCD has to choose a faithfulness constraint. By specificity, the choice boils down to the two positional constraints F_1^{pos} and F_2^{pos}. Crucially, they have the same autonomy, since they both are winner-preferring in an ERC where the corresponding general faithfulness constraint is the only winner-preferring constraint. Thus, LFCD has no way to make a principled choice between the two positional faithfulness constraints. And if it chooses F_1^{pos} at the second iteration, it will return an unrestrictive ranking.

The comparison with (19) shows that LFCD has succeeded in the Korean case (18) only "by accident". The Korean pattern happens not to contain the form [da]. It is for this reason that F_2^{pos} "looks" more autonomous than F_1^{pos}. But the presence or the absence of this form [da] has nothing to do with the reason why the positional faithfulness constraint F_2^{pos} needs to be ranked above the other positional faithfulness constraint F_1^{pos}. Thus, LFCD effectively happens to learn the correct relative ranking of the two positional faithfulness constraints in the Korean case for a reason which has nothing to do with why that ranking is actually needed. The weakness of the algorithm can be pinpointed by looking at a variant of the Korean pattern, which requires that same ranking condition between the positional faithfulness constraints, but does not display the fortuitous condition (the absence of [da]) that had helped the algorithm in the Korean case. The language (19) is indeed such a case.

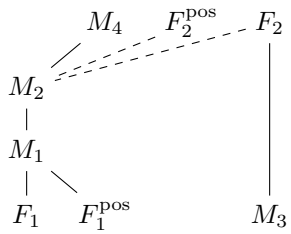
BCD fails on six out of ten of the languages of type IV in the voicing×aspiration typology. The reason for the failure is the same across all six languages. To illustrate the nature of the challenge, consider the language which only consists of [t$^{\text{h}}$a], [at$^{\text{h}}$], [ada] and [at$^{\text{h}}$a] (plus the unmarked forms). The corresponding

ERC set is provided in (20a). This language is generated by a total ranking iff it is a refinement of the partial ranking (20b). The two dashed lines converging on M_2 represent the disjunctive ranking condition that either F_2 or F_2^{pos} be ranked above M_2. In the former case, the language counts as of type IV, as it requires the general faithfulness constraint F_2 to be ranked above the general faithfulness constraint F_1.

(20) a.

	F_1	F_2	F_1^{pos}	F_2^{pos}	M_1	M_2	M_3	M_4
([/t$^{\text{h}}$a/, t$^{\text{h}}$a], ~~[ta]~~)		W		W			L	
([/t$^{\text{h}}$a/, t$^{\text{h}}$a], ~~[da]~~)	W	W	W	W	W		L	
([/t$^{\text{h}}$a/, t$^{\text{h}}$a], ~~[d$^{\text{h}}$a]~~)	W		W		W			W
(/at$^{\text{h}}$/, [at$^{\text{h}}$], ~~[at]~~)		W					L	
(/at$^{\text{h}}$/, [at$^{\text{h}}$], ~~[ad]~~)	W	W			W		L	
([/at$^{\text{h}}$/, at$^{\text{h}}$], ~~[ad$^{\text{h}}$]~~)	W				W			W
(/ada/, [ada], ~~[ata]~~)	W		W		L	W		
(/ada/, [ada], ~~[at$^{\text{h}}$a]~~)	W	W	W	W	L	W	W	
(/ada/, [ada], ~~[ad$^{\text{h}}$a]~~)		W		W			W	W
(/at$^{\text{h}}$a/, [at$^{\text{h}}$a], ~~[ata]~~)		W		W			L	
(/at$^{\text{h}}$a/, [at$^{\text{h}}$a], ~~[ada]~~)	W	W	W	W	W	L	L	
(/at$^{\text{h}}$a/, [at$^{\text{h}}$a], ~~[ad$^{\text{h}}$a]~~)	W		W		W	L		W

b.



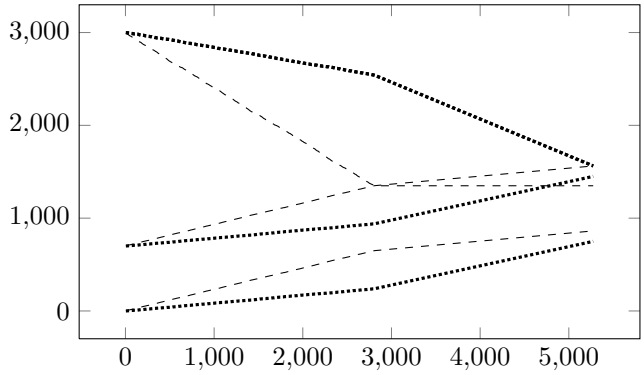
At the first iteration, BCD chooses the markedness constraint M_4 and simplifies the ERC set accordingly. At the second iteration, it needs to choose a set of specific faithfulness constraints which free up some markedness constraints. The positional faithfulness constraint F_1^{pos} frees up two markedness constraints, namely M_1 and M_2. The positional faithfulness constraint F_2^{pos} instead only frees up M_2. BCD thus chooses the positional constraint F_1^{pos}, as it frees up more markedness constraints than the other positional constraint F_2^{pos}. This choice is incorrect, because it leads to a ranking which assigns F_1^{pos} above M_1 and thus fails at restrictiveness according to the minimal ranking conditions (20b). This example shows that BCD's greediness for freeing up as many markedness constraints as possible can lead to unrestrictive rankings.

The CEDRA succeeds on each of the ten languages of type IV in the voicing×aspiration typology. To illustrate, I provide in (21a) the final ranking values learned when trained on the Korean pattern (18a). As these ranking values capture all the ranking conditions required in (18b), the algorithm has succeeded. The corresponding ranking dynamics is plotted in (21b): the ranking values of the constraints F_1, F_1^{pos}, M_1 and M_2 are plotted with a dotted line (the lines corresponding to M_1 and M_2 completely overlap). The ranking values corresponding to F_2, F_2^{pos}, and M_3 are plotted with a dashed line; the ranking value of M_4 is not plotted (it stays put at its initial ranking value).

(21) a.

M_4	3000.04
F_2^{pos}	1562.66
M_2	1561.77
M_1	1561.67
F_1^{pos}	1450.41
M_3	1349.03
F_2	862.66
F_1	750.41

b.



See Magri (2014) for an analysis of this ranking dynamics, and an explanation of how the CEDRA succeeds on the languages of type IV in the two typologies. Again, here I only want to underscore the fact that the CEDRA could not succeed if it performed no constraint promotion, because it would be unable to pull apart the faithfulness constraints and learn their relative ranking.

6 Conclusions

The problem of *pure phonotactic learning* in OT is as follows: given a set of surface forms, find a ranking which is *consistent* (i.e., the given set of forms belongs to the set of forms licit according to that

ranking) and is furthermore *restrictive* (i.e., no other consistent ranking generates a subset of licit forms). From a computational perspective, this problem formulates the classical *Subset problem* (Fodor & Sakas 2005) within OT phonology. From a modeling perspective, it formalizes the task faced by a child acquiring phonotactics at an early developmental stage (Hayes 2004). As this problem has been shown to be intractable (Magri 2013), we need to develop heuristic approaches and to explore their modeling implications.

Hayes (2004) and Prince & Tesar (2004) outline one such heuristic approach. The core idea of the approach is to bias Tesar & Smolensky's (1998) CD algorithm in such a way that it chooses a markedness over a faithfulness constraint at every iteration where that is possible. And chooses a proper faithfulness constraint, whenever no markedness constraints can be chosen. Crucially, CD is a *batch* algorithm: it constructs the ranking iteratively, by looking at the entire batch of data. The modifications of CD proposed by Hayes and Prince and Tesar take advantage of this fact: their criteria for the choice of the constraint to be ranked at the current iteration crucially take into account the entire batch of data. Because of the full strength of the batch learning scheme, this approach to restrictiveness plays the role of *Goliath*.

Hayes and Prince and Tesar conjecture that the full strength of the batch approach is needed to succeed at pure phonotactic learning. They thus dismiss the alternative *error-driven* learning scheme as too weak. An error-driven learner performs a sequence of instantaneous re-rankings based only on the current piece of data, without ever being able to compute the implications of the current re-ranking for the entire batch of data. Furthermore, the only provision towards restrictiveness consists of a restrictive initial grammar, which could nonetheless be unduly enlarged as the learner is pulled around by the sequence of instantaneous re-rankings. Because of its limited resources, the error-driven approach plays the role of *David*.

In this paper, I have compared the two approaches on two OT typologies, that yield a total of roughly seventy languages. I have considered both Hayes' and Prince and Tesar's implementations of the batch approach, and I have shown that they fail on a number of cases. I have then considered an implementation of the error-driven approach which has three crucial properties. First, it starts from a proper initial ranking, whereby the markedness constraints are ranked above the positional faithfulness constraints which are in turn ranked above the general ones. Second, it performs constraint demotion as well as promotion. Because of the promotion component, it is able to re-rank and pull apart the faithfulness constraints throughout learning. Third, it is trained in the idealized regime of uniform sampling, whereby the data are uniformly sampled and fed to the learner. I have shown that this implementation of error-driven learning succeeds on each language in the two typologies considered. In other words, *David defeats Goliath*.

References

Boersma, Paul (1998). *Functional Phonology*. Ph.D. thesis, University of Amsterdam, The Netherlands. The Hague: Holland Academic Graphics.

Fodor, Janet Dean & William Gregory Sakas (2005). The Subset Principle in syntax: costs of compliance. *Linguistics* 41, 513–569.

Hayes, Bruce (2004). Phonological acquisition in Optimality Theory: The early stages. Kager, René, Joe Pater & Wim Zonneveld (eds.), *Constraints in Phonological Acquisition*, Cambridge University Press, Cambridge, 158–203.

Hayes, Bruce, Bruce Tesar & Kie Zuraw (2003). OTSoft 2.3. Software package available at http://www.linguistics.ucla.edu/people/hayes/otsoft/.

Jusczyk, P. W., A. D. Friederici, J. M. I. Wessels, V. Y. Svenkerud & A. Jusczyk (1993). Infants' sensitivity to the sound patterns of native language words. *Journal of Memory and Language* 32, 402–420.

Lombardi, Linda (1999). Positional faithfulness and voicing assimilation in Optimality Theory. *Natural Language and Linguistic Theory* 17, 267–302.

Magri, Giorgio (2012). Convergence of error-driven ranking algorithms. *Phonology* 29:2, 213–269.

Magri, Giorgio (2013). The complexity of learning in OT and its implications for the acquisition of phonotactics. *Linguistic Inquiry* 44.3, p. 433468.

Magri, Giorgio (2014). The "markedness-above-faithfulness" approach to restrictiveness: foundations and implementations. Manuscript, CNRS and UiL-OTS.

Prince, Alan (2002). Entailed ranking arguments. Ms., Rutgers University, New Brunswick, NJ. Rutgers Optimality Archive, ROA 500. Available at http://www.roa.rutgers.edu.

Prince, Alan & Bruce Tesar (2004). Learning phonotactic distributions. Kager, R., J. Pater & W. Zonneveld (eds.), *Constraints in Phonological Acquisition*, Cambridge University Press, 245–291.

Tesar, Bruce (2013). *Output-Driven Phonology: Theory and Learning*. Cambridge Studies in Linguistics.

Tesar, Bruce & Paul Smolensky (1998). Learnability in Optimality Theory. *Linguistic Inquiry* 29, 229–268.

Wexler, Kenneth & Peter W. Culicover (1980). *Formal Principles of Language Acquisition*. MIT Press, Cambridge, MA.

Yanovich, Igor (2012). The logic of OT rankings. MIT manuscript.